Due: Wednesday, January 26, beginning of lecture

NOTE: Each problem set only counts 5% of your mark, but it is important to do your own work (but see below). These assignments will be followed by term tests, each worth 15% of your final grade. You may consult with others concerning the general approach for solving problems on assignments, but you must write up all solutions entirely on your own. You may chose to work in pairs and then submit one assignment but both partners should work on all questions. Anything else is *plagiarism*, and is subject to the University's Code of Behavior. You will receive 1/5 points for any question/subquestion for which you say "I do not know how to answer this question". You will receive .5/5 points if you just leave the question blank.

1. Consider the problem of *interval colouring* where the goal is to minimize the number of colours used in a feasible colouring where a feasible colouring is one in which intervals receiving the same colour do not intersect.

   Consider the following greedy algorithm for the interval colouring problem:

   ```
   Sort $\mathcal{I}$ so that $s_1 \leq s_2 \ldots \leq s_n$
   For $j = 1..n$
           Colour interval $I_j$ with the lowest numbered colour $c$ such that
           for any $k < j$, if $I_k$ and $I_j$ intersect then $I_k$ did not receive colour $c$.
   EndFor
   ```

   (a) Prove that this greedy algorithm is optimal. Hint: what can be said at the time that the algorithm first uses it largest colour?

   (b) Show that the algorithm is not optimal if the intervals are not first sorted? Would the algorithm be optimal if the intervals were sorted by non-decreasing (or non-increasing) processing times?

2. Consider the following "best fit" greedy algorithm for the unweighted interval scheduling problem on m identical machines.

   ```
   Sort $\mathcal{I} = \{I_1, \ldots, I_n\}$ so that $f_1 \leq f_2 \leq \ldots f_n$
   For $j = 1..m$
       $t_j := 0$   % $t_j$ will be the (current) latest finish time on machine $j$
   EndFor
   For $i = 1..n$
       If there exists $j$ such that $t_j \leq s_i$ then
           schedule $I_i$ on that machine $j$ which minimizes $s_i - t_j \geq 0$;
           $t_j := f_i$
       EndIf
   Endfor
   ```

   (a) Prove that "best fit" always produces an optimal schedule.

(b) Consider the "first fit" greedy algorithm where we replace the statement "schedule I_i on that machine j which minimizes $s_i - t_j \geq 0$" by "schedule I_i on the machine of smallest index j such that $s_i - t_j \geq 0$".
Show that "first fit" is not an optimal algorithm.

(c) What is the best approximation ratio you can prove for "first fit"?

3. Consider the following scheduling problem. We are given n jobs $J_1, \ldots, J_n$ where each job J_i is described by a processing time p_i and a value v_i. A schedule is simply an ordering π of the jobs so that $J_{\pi(1)}$ is scheduled first, $J_{\pi(2)}$ next, etc. The goal is to schedule all jobs sequentially (without overlap) so as to minimize the weighted completion time $\sum_{1 \leq i \leq n} v_{\pi(i)} * C_{\pi(i)}$ where $C_{\pi(1)} = p_{\pi(1)}$ and for $i \geq 2$, $C_{\pi(i)} = C_{\pi(i-1)} + p_{\pi(i)}$ is the completion time of the i^{th} job in the schedule. Give a (fixed order) greedy algorithm which always produces an optimal schedule. Prove that your algorithm is optimal.
Hint: Study the exchange argument used in section 4.2.

4. (a) Consider the unweighted interval scheduling problem and the third non-optimal greedy algorithm given in the text; namely in each iteration select an interval $I \in \mathcal{I}$ that intersects the fewest intervals in $\mathcal{I}$. Then remove that I and all the intervals it intersects. Give an argument showing that this algorithm is a 2-approximation algorithm.
Hint: Want to show that when we choose an interval I, there can be at most 2 disjoint intervals intersecting I.

(b) The text gives an example where this algorithm accepts three intervals whereas an optimal algorithm would accept 4 intervals showing that the approximation ratio is at best 4/3. Can you improve either the upper (2) or lower (4/3) bound for the approximation ratio of this algorithm?

5. Consider the following $\{0, 2, 3\}$ variant of the knapsack problem. Given n items $(v_1, w_1), \ldots, (v_n, w_n)$ and knapsack weight bound W, we define a feasible solution σ as a sequence $\sigma =< c_1, \ldots, c_n >$ in $\{0, 2, 3\}^n$ such that $\sum_i c_i * w_i \leq W$ and define its value $V(\sigma)$ as $\sum_i c_i * v_i$. That is, every item that occurs in the knapsack occurs 2 or 3 times.

Describe a dynamic programming algorithm for this $\{0, 2, 3\}$ knapsack problem by defining an appropriate "semantic array" A , and a recurrence (including the base cases) for computing elements of A. Justify why the recurrence is correct.

6. Possibly more to follow